Post-processing and visualisation of large-scale DEM simulation data with the open-source VELaSSCo platform

Post-processing and visualisation of large-scale DEM simulation data with the open-source VELaSSCo platform

Journal Title
XX(X):1–12

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

John P. Morrissey[1], Prabhat Totoo[1], Kevin J. Hanley[1], Stefanos-Aldo Papanicolopulos[1], Jin Y. Ooi[1], Iván Cores Gonzalez[2], Bruno Raffin[2], Seyedmorteza Mostajabodaveh[3] and Thomas Gierlinger[3]

Abstract

Regardless of its origin, in the near future the challenge will not be how to generate data, but rather how to manage big and highly distributed data to make it more easily handled and more accessible by users on their personal devices. VELaSSCo (**V**isualization for **E**xtremely **La**rge-**S**cale **S**cientific **Co**mputing) is a platform developed to provide new visual analysis methods for large-scale simulations serving the petabyte era. The platform adopts Big Data tools/architectures to enable in-situ processing for analytics of engineering and scientific data and hardware-accelerated interactive visualisation.

In large-scale simulations, the domain is partitioned across several thousands of nodes, and the data (mesh and results) is stored on those nodes in a distributed manner. The VELaSSCo platform accesses this distributed information, processes the raw data and returns the results back to the users for local visualisation by their specific visualisation clients and tools. The global goal of VELaSSCo is to provide Big Data tools for the engineering and scientific community, in order to better manipulate simulations with billions of distributed records. The ability to easily handle large amounts of data will also enable larger, higher resolution simulations which will allow the scientific and engineering communities to garner new knowledge from simulations previously considered too large to handle. This paper shows, by means of selected Discrete Element Method (DEM) simulation use cases, that the VELaSSCo platform facilitates distributed post-processing and visualisation of large engineering data sets.

Keywords

discrete element method, DEM, data analytics, discrete-to-continuum, visualisation, Apache Hadoop, HBase

Introduction

The Discrete Element Method (DEM) is a simulation tool used in engineering to model complex systems of particulates at the particle scale by specifying a relatively small number of microstructural parameters. DEM is very closely related to molecular dynamics (MD), an analysis tool used in chemistry, biochemistry and materials science[1]. The largest difference between DEM and MD is the scale of interest: MD simulates the interactions between individual atoms or molecules, whereas DEM is used to simulate soils, powders and grains at much larger scales. The fundamental algorithm for MD was proposed in the 1950s[2,3] with the related DEM methodology developed later in the 1970s[4]. Since then, DEM has become increasingly popular for analysing the particle-scale mechanisms that underlie the complexity of the overall material response. In the most common implementation of DEM, particles are modelled as rigid bodies with finite size, inertia and stiffness. Deformations of particles at the contact points are captured by permitting overlaps between the interacting bodies. The particle sizes usually range from microns to tens of millimetres. A timestepping algorithm is implemented and, at each successive timestep, interparticle forces are evaluated at the contact points using suitable force–displacement relations. The resultant force on each particle is calculated by summation. Newton's Second Law is applied to determine particle accelerations based on the resultant forces[5] which are integrated numerically to find particle velocities and displacements; hence, positions may be updated during each timestep.

DEM is capable of providing extremely detailed information about the microstructure of a simulated material and its temporal evolution. Such information is of interest to researchers working with a broad range of granular materials from minerals to pharmaceuticals. Although some barriers remain to the adoption of DEM for industrial applications[6], its popularity continues to grow[1,7]. There has been an approximately linear rate of increase in the number of DEM-related papers published since 1996[5]. The growing interest in DEM among the scientific community is further emphasised by the publication of a number of special issues

[1]School of Engineering, Institute for Infrastructure and Environment, The University of Edinburgh, Edinburgh EH9 3JL, UK
[2]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
[3]Fraunhofer IGD, Fraunhoferstraße 5, 64283 Darmstadt, Germany

Corresponding author:
John P. Morrissey, School of Engineering, The University of Edinburgh, Thomas Bayes Road, The King's Buildings, Edinburgh, EH9 3FG, UK.
Email: J.Morrissey@ed.ac.uk

of journals devoted to DEM within the past decade, e.g., Eng. Computation. 26(6); Granul. Matter 11(5); Powder Technol. 193(3); Powder Technol. 248.

This interest in DEM has been driven by the increasing availability of substantial amounts of computational power, enabling fully three-dimensional simulations to be run which are of practical use. MPI-parallelised solvers such as LAMMPS[8] and LIGGGHTS[9] scale to hundreds of nodes and beyond on clusters. The popular commercial codes PFC[10] and EDEM[11] use parallel processing to exploit modern multi-core architectures. Inevitably, increases in computational power and the exploitation of parallelism lead to commensurate increases in the amount of data generated by particle simulations. Although predictions that a 10 million particle problem would be considered 'easy' by 2011[12] were optimistic, the predicted continual growth of simulation size has been proved correct. The accessible timescales and number of atoms in MD simulations have closely tracked Moore's Law[13]. A large DEM simulation now generates >1 TB of data (information about each particle and each contact in the system) with hundreds of MB of data per timestep. For large-scale N-body simulations, the current limitation is data post-processing which is often done on a single workstation. This is unsustainable as the data storage requirements and CPU effort required for analyses become ever-larger.

Post-processing of particle simulations is often essential for correct and/or meaningful interpretation of the data. In particular, consider the discrete-to-continuum (or Discrete2Continuum/D2C) transformation. This applies temporal and spatial coarse graining methods[14,15] to DEM simulation data in order to compute bulk quantities such as solid fraction or stress tensors that are projected onto a continuum field. A snapshot of the data at one instant appears chaotic and does not show the concealed, underlying trends that emerge over longer timescales. This computationally expensive post-processing allows the long-term behaviour of scientific interest to be distinguished from short-term random and transient fluctuations. The current limitations on processing of large-scale simulations mean that often analyses are conducted with small subsets of the available data. This leads to two related problems: firstly, restricting the analyses to a small fraction of the data could yield misleading results, and secondly, some of these data that are discarded without analysis could contain results of scientific interest.

There are several papers in the literature showing the usefulness of the D2C transformation. Weinhart et al.[16] recently simulated silo discharge using DEM and applied spatio-temporal coarse graining to the data. They quantified stresses from interparticle contact forces and demarcated three distinct flow regions by comparing macroscopic fields obtained from spatial averaging. This analysis identified stagnant, highly stressed regions adjacent to the silo outlet. This indicates that, in order to achieve mass flow where the stored solids are all flowing concurrently, the silo would need to be redesigned, for example, by changing the hopper angle. Clark et al.[17] applied coarse graining to an impeller-driven mixer system to evaluate local densities, averaged velocities and granular temperatures. The quality of mixing may be evaluated using this information; the extent to which materials are mixed can significantly affect product quality[18] and sometimes safety, e.g., excipients and active pharmaceutical ingredients (APIs) must be blended thoroughly before tableting to ensure that no tablet contains a potentially dangerous surfeit of API.

Such analyses are very computationally expensive and were formerly limited to small data sets with few particles and/or timesteps. This was due to the lack of a suitable parallelised tool for distributed analysis of scientific Big Data. In this paper, we discuss the VELaSSCo software platform aiming to address this problem[19]. This platform is developed within an EC FP7-funded project, VELaSSCo: "Visualisation for Extremely Large-Scale Scientific Computing". VELaSSCo is a three-year project which ran from January 2014 to December 2016. The goal of VELaSSCo was to create the VELaSSCo platform for distributed post-processing and visualisation of very large engineering data sets[20]. These data sets include DEM, the finite element method (FEM)[21–23] and computational fluid dynamics (CFD)[24–26]. VELaSSCo is a consortium of seven European partners: The University of Edinburgh (UK), CIMNE, Atos Research & Innovation (both Spain), INRIA (France), Fraunhofer IGD (Germany), SINTEF and Jotne EPM (both Norway).

The development of the VELaSSCo platform has been guided by a user panel. The panel was consulted at the start of the project to learn of their requirements for the platform, they were kept apprised of developments and they evaluated both an early prototype of the platform and the final platform. User feedback on the prototype has been invaluable to inform its subsequent development. The user panel had around 60 members, with approximately a 70:30 balance between academia and industry. Usability testing in VELaSSCo followed a *'goal, question, metric'* or GQM approach to software metrics[27]. Two variants of the VELaSSCo platform have been created: a fully open-source version and a proprietary version which use Apache HBase and EXPRESS Data Manager™ (EDM) as database systems, respectively. Only the open-source version is described in this paper.

The aim of this paper is to show, by means of selected use cases, how the VELaSSCo platform facilitates distributed post-processing and visualisation of large engineering data sets. A comprehensive overview of the platform is provided along with a description of the D2C transformation for DEM data. Some practical guidance is provided for using the platform, while two DEM use cases demonstrate the capabilities of the VELaSSCo platform.

Overview of the VELaSSCo Platform

The VELaSSCo platform was developed to work both with open-source and closed-source software. The open-source version is built on top of the open-source Hadoop software stack: a Java-based framework for distributed storage and processing of big data. The software components within the Hadoop framework include the Hadoop Distributed File System (HDFS) which stores files in a distributed, split and redundant way for parallel access; HBase, a distributed database built on top of HDFS providing a transparent means for storing and accessing data on the cluster; YARN which

manages the computing resources and schedules jobs across the cluster; and MapReduce which refers to the programming model for large-scale data processing. Figure 1 shows the three main layers in the architecture: the visualisation clients, the Core Analytics Module and the Data Layer.

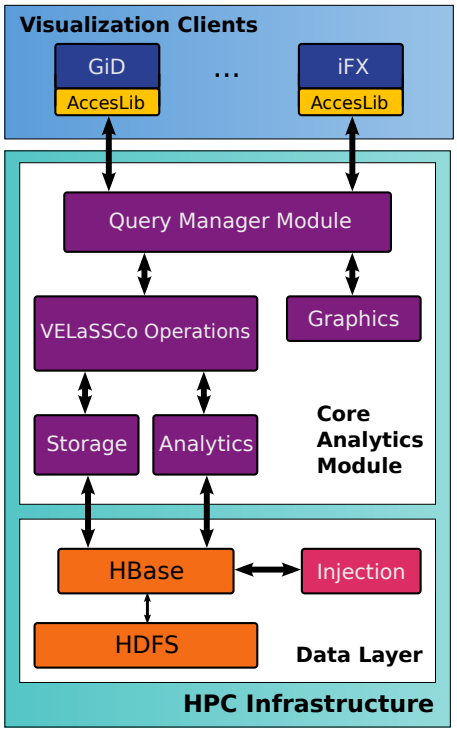


Figure 1. VELaSSCo open-source architecture

Visualisation Clients

The visualisation client triggers queries through the access library which is the interface between the client and the platform. The complex D2C query is explained in a later section. The visualisation clients (currently two: GiD[28] and iFX[29]) implement a plugin to access the platform. Integration between components is done through a Thrift API, so the plugin implements a public API provided by the platform. The results output by the queries are returned to the clients through an internal VELaSSCo format which is designed to minimize latencies through high-speed transfer. The following requirements were considered when designing the format:

- **Minimal data size**. Although the headers, separators, and metadata make parsing and conversion easier, they increase file size. To reduce the file size, they are minimized.
- **Binary data**. Since data are being transferred over a network, the payload data are in binary form to avoid conversion from an ASCII representation.
- **Processing friendly layout**. The stored data are directly accessible by processing units without any conversion. The payload can be uploaded to OpenGL/D3D buffers directly. While the GPU is the typical target, this also applies to CPUs.

In this format, a batch of data is represented as a file. A file is a self-contained binary 'blob' of data that includes all the information to display a scene. Such a file consists of two sections, a header section and a data section. The header contains a description of how the payload is handled. The data section is simply a collection of buffers. Within a C/C++ program a file could be accessed as the struct shown in Listing 1.

```
struct File
{
 // file information
 struct Header
 {
  // identification as VELaSSCo data
  uint8_t  magic[8];                // magic number ("VELaSSCo", UTF-8)
  uint32_t version;                 // version number (100 = V1.00)

  // information about the content
  uint64_t descriptionBytes;        // Size: description block (B)
  uint64_t metaBytes;               // Size: meta block (B)

  // sizes of the contained buffers
  uint64_t vertexDefinitionsBytes;  // Size: vertex definition buffer (B)
  uint64_t vertexAttributesBytes;   // Size: vertex attributes buffer (B)
  uint64_t edgeDefinitionsBytes;    // Size: edge definition buffer (B)
  uint64_t edgeAttributesBytes;     // Size: edge attributes buffer (B)
  uint64_t faceDefinitionsBytes;    // Size: face definition buffer (B)
  uint64_t faceAttributesBytes;     // Size: face attributes buffer (B)
  uint64_t cellDefinitionsBytes;    // Size: cell definition buffer (B)
  uint64_t cellAttributesBytes;     // Size: cell attributes buffer (B)
 };

 struct Data
 {
  // description of file content
  uint8_t* description;

  // additional data
  uint8_t* meta;

  // vertex block
  uint8_t* vertexDefinitions;
  uint8_t* vertexAttributes;

  // edge block
  uint8_t* edgeDefinitions;
  uint8_t* edgeAttributes;

  // face block
  uint8_t* faceDefinitions;
  uint8_t* faceAttributes;

  // cell block
  uint8_t* cellDefinitions;
  uint8_t* cellAttributes;
 };
};
```

Listing 1. The internal VELaSSCo platform file format. It is used to transfer DEM/FEM meshes from the HPC infrastructure to visualisation clients.

For example, transferring particles with information requires the use of vertex definitions and vertex attributes. The vertex definitions buffer keeps particle centres along with vertex attributes which store their attributes (e.g., radius). In addition, pressure, velocity, etc. can be packed into the file format sent to the client. The description buffer is responsible for implicitly informing the visualisation clients about incoming file contents and how they should be interpreted.

In case of progressive visualisation, e.g., visualising particles across different timesteps as an animation, this format does not explicitly support streaming or level-of-detail display. The intention is that in such cases data are not sent as a single file, but rather as a stream of files (which are transferred one after another). Later, files could either contain new data that replace old data (e.g., a new set of triangle strips), but they could also provide incremental data. For example, in case of progressive meshes, new files could add new vertices and edges that are used to expand available geometries to a new detail level. How new data are to be interpreted could be defined in the description buffer.

The rendering algorithms are optimized to provide highest efficiency during rendering of the received data. In DEM cases in which the particles need to be drawn, they are

received in a compact format. As mentioned before, the client can redirect it to the rendering module without any conversion. The iFX visualisation client supports two rendering modes: rasterization-based and ray-tracing-based. In both cases, particle data are sent to the rendering device without any conversion. Furthermore, the deferred approach[30] is utilized to improve massive mesh (e.g., very high number of particles) rendering efficiency. In addition to position, material, etc. buffers stored as G-Buffer, for each attribute, a buffer is filled during the first stage. Although rendering simulation meshes consisting of cells is done via triangulation, a more efficient approach is utilized for particle rendering. In the first approach, the GPU's pipeline is programmed so it produces a sprite in the geometry shader for each particle. Spheres inside every sprite are ray casted in the fragment shader[31]. To improve the final result, screen space techniques such as depth darkening[32] and ambient occlusion[30] can be applied to the shaded image to improve the final image's visual perception quality. In the ray-tracing-based approach, the received data are used to produce a bounding box for each particle; then a Bounding Volume Hierarchy[33] acceleration structure is created and used for speeding up ray tracing. To get pixel-accurate results, instead of triangulating the spheres, ray-sphere intersection is done analytically. In addition, path-tracing[34] and ray-tracing-based ambient occlusion can be utilized to give better perceptual clues about depth, curvature and proximity. In the second stage of deferred shading, attribute values are mapped to a colour ramp and used during lighting calculation as the geometry colours to make the final images easier to analyse.

In addition, the mesh data transferred to the client can be post-processed by existing tools in visualisation clients. For example, discrete-to-continuum result meshes can be cut through, or streamlines can be computed on the client side to help the user get a better understanding of attribute changing behaviours inside the mesh.

Core Analytics Module

The queries triggered by the visualisation clients are processed by the Core Analytics Module which retrieves and analyses the data. This module runs on the HPC infrastructure and delivers the query results to the visualisation clients. The data are compressed to avoid overloading the network. Different internal modules are in charge of the different functions:

- **Query Manager Module.** This module is in charge of receiving and decomposing the queries into small and simple queries or operations. These operations can be reused by different queries, facilitating the addition of new queries to the platform. Basically it acts as a multi-user server that is able to process multiple queries at the same time.
- **VELaSSCo Operations.** It provides an abstraction of all the operations. These operations are shared between the open- and closed-source versions of the platform.
- **Storage.** It is in charge of delivering the results of non-computationally-intensive operations. This module is directly connected to HBase to read the data without any kind of analytic computation. An example might be an operation to return the velocity value from a particular particle's ID.
- **Analytics.** It executes the computationally-intensive calculations to produce the post-processing results. These calculations are executed as MapReduce jobs distributed over the Hadoop nodes. This module also stores the results of the most useful queries in HBase, so when the users execute these queries for the first time, the results are calculated and stored, providing near-instant results when the query is next called.

Data Layer

The Data Layer is the deepest layer which stores and manages the data. The core of this layer is HDFS which is responsible for data storage and distribution across regions. It guarantees data replication across the Hadoop cluster and fault tolerance. HBase is a non-relational database that works on top of HDFS and provides real-time access to the data. As it stores information with a key/value structure, all data can be easily accessed.

The simulation data can be received by the system in real-time or via files with the final data stored on disk. The injection module manages both data sources. Apache Flume is used to ingest simulation data into HBase. Once the data are stored in HBase, the injection module is able to perform some post-injection actions, e.g., execute a set of frequently called queries and store the results data to provide faster results to users in future queries.

Data are provided to the platform in a file format which has been developed as an extension to ISO 10303-209:2014 *Application protocol:"Multidisciplinary Analysis and Design"*[35,36]. A well-defined input format allows the platform to be used by many different simulation solvers with ease. An example of the standard input for DEM data for spherical particles is presented in Listing 2, which also shows how an optional variable, angular velocity (in a vector format), can be included. Additional variables can be defined as either vectors or scalars. The orientation of non-spherical particles can be accounted for by including the nine-component orientation matrix as the first additional output in the particle file.

Listing 2. File format for DEM particle data

```
TIMESTEP PARTICLES
0.02 11
ID GROUP TYPE VOLUME MASS PX PY PZ VX VY VZ AngVel_X AngVel_Y AngVel_Z
1 1 1 4.18879e-6 0.010472 0.015492 0.016146 0.0008229 0 0 0.19618 0 0 0
5 2 1 4.18879e-6 0.010472 0.016643 0.019136 0.0092912 0 0 0.19618 0 0 0
.......

TIMESTEP PARTICLES
0.04 44
ID GROUP TYPE VOLUME MASS PX PY PZ VX VY VZ AngVel_X AngVel_Y AngVel_Z
1 1 1 4.18879e-6 0.010472 0.015492 0.016146 0.0008229 0 0 0.19618 0 0 0
5 2 1 4.18879e-6 0.010472 0.016643 0.019136 0.0092912 0 0 0.19618 0 0 0
.......
```

Both particle–particle and particle–geometry contact data follow the same style and examples of each file can be seen in Listing 3 and Listing 4, respectively. Geometry meshes can be imported as standard STL files.

Listing 3. File format for DEM particle–particle contact data

```
TIMESTEP CONTACTS
0.02 6
P1 P2 CX CY CZ FX FY FZ
11 1 0.004 −0.0055 0.0005 0.727312 −0.098406 2.70531
10 7 0.009 −0.0055 0.0005 −0.00396415 0.235619 0.199911
.......

TIMESTEP CONTACTS
0.04 1
P1 P2 CX CY CZ FX FY FZ
11 1 0.004 −0.0055 0.0005 0.727312 −0.098406 2.70531
```

Listing 4. File format for DEM particle–geometry contact data

```
TIMESTEP CONTACTS
0.02 4
P1 WALL CX CY CZ FX FY FZ
10 1 −0.198716 −0.0265078 0.087761 −0 −0 0.00993776
11 1 −0.0178762 0.245043 3.74038 −0 −0 0.00993035
.......

TIMESTEP CONTACTS
0.04 7
P1 WALL CX CY CZ FX FY FZ
10 1 −0.0106519 −0.238474 0.0431609 −0 −0 0.00994131
11 1 0.225941 0.0361603 0.285362 −0 −0 0.00994407
.......
```

Analysis of DEM Simulation Data with the VELaSSCo Platform

The VELaSSCo platform has been designed such that using VELaSSCo for analysis of data becomes a relatively trivial task for the end-user, with the complexity of implementation hidden in the background. The user accesses the platform through a simple GUI which allows manipulation of the data using the features of the visualisation client. Currently plug-ins have been written for the popular iFX and GiD visualisation clients that allow these to work seamlessly with the VELaSSCo platform.

The Discrete2Continuum (D2C) transformation is one of the advanced analysis tools that has been incorporated within the platform. It applies temporal and spatial coarse graining methods to DEM simulation data to compute bulk quantities (such as solid fraction or stress tensors) that are projected onto a continuum field. The interface to the D2C query is presented in Figure 2. The interface displays the many options related to the transformation in an easy-to-access manner for the end-user in the form of a simple GUI.

The processing option can be divided into four main sections:

- **Selection of processing time range:** This option gives the end user the ability to process data from a specific time range. This can be selected as all timesteps present, an inclusive range of start and stop times, or an interval for an inclusive range of start and stop times (i.e., every tenth timestep between x and y).
- **Spatial averaging (coarse graining) parameters:** These are the options for the D2C transformation weighting function. The user has a choice between a simple Heaviside function and a Gaussian weighting function. The key parameter here is the coarse graining width, which can typically be related to the particle size in quasi-static simulations [15].
- **Temporal averaging parameters:** Temporal averaging allows the end user to analyse the results over a timeframe that is relevant to the problem being studied – the data output frequency from the DEM solver is not always the most relevant.
- **Spatial integral parameters:** This is a novel function that allows certain problem types to be considered as plane strain (two-dimensional) problems, reducing the computational cost associated with the D2C transformation.

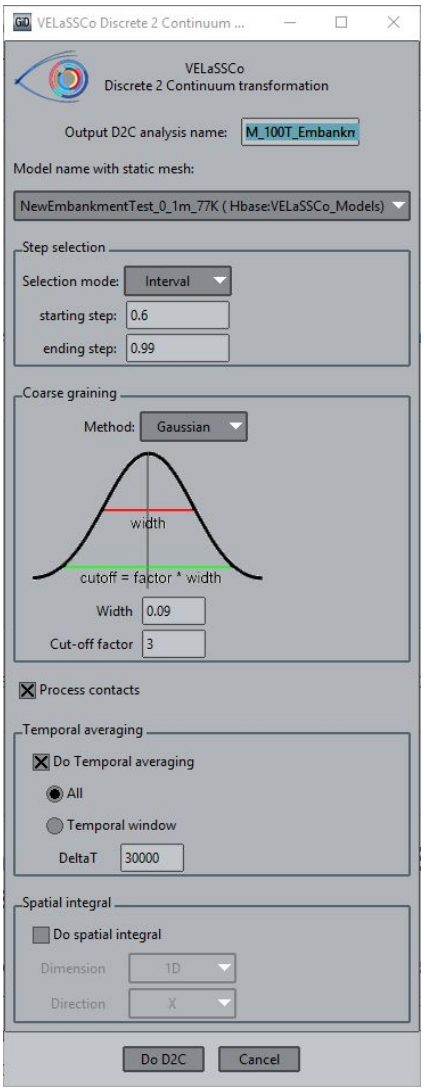


Figure 2. Discrete2Continuum interface on VELaSSCo

Experimental Setup and Performance Evaluation

The VELaSSCo platform is deployed on a subset of the Eddie cluster managed by the University of Edinburgh Computer and Data Facility (ECDF). The setup used for the experiments consists of 40 nodes; each node comprises two Intel Xeon CPU E5-2630 @ 2.40 GHz (16 cores) with 64 GB of RAM and 2.8 TB of local disk storage. Without HBase replication of ingested data, this setup allows storage and processing of simulations of up to 112 TB in size. There is an additional 50 TB of network storage used to store simulation files for ingestion.

Utilising the distributed nature of the data allows significant performance gains to be made. In particular,

the default HBase replication factor ensures data blocks are distributed across multiple regions which improves load balancing during a job execution. Data for one timestep correspond to one data block. During ingestion, a distribution strategy can be specified, e.g., random timesteps/blocks allocated to random nodes or a range of consecutive timesteps allocated to random nodes. The latter may not be the best for parallel execution. Sometimes manual data relocation is performed to have an optimal distribution of data for a specific simulation.

The correctness of the algorithms has been thoroughly checked against a number of unit tests that take in a set of input parameters and expected outputs. The results have also been compared with those produced by other tools such as Particle Analytics software [37]. The robustness of the algorithm was tested using a fault injection strategy whereby a set of invalid input parameters or simulation data values were used in order to ensure that exceptional cases are handled in an appropriate manner in the algorithm without failing or creating inconsistent data in the Data Layer. For the efficiency and scalability evaluation, a set of benchmarks consisting of the two use cases described in the later sections were defined in order to assess the performance by running a number of different trials. The parameters taken into account for reproducing different benchmark test sets were the number of processors/nodes used for the computation and/or the size of the input data. Measuring the spent time of an algorithm that uses several Hadoop machines in parallel is not a trivial task. Since the input of the algorithm is stored in HBase, instead of excluding nodes from the computation by decommitting them through the editing of the configuration file in Hadoop, the number of splits of HBase regions was forced during the experiment in order to create a number of mappers equal to the number of nodes that were required for use in parallel during the experiments. For these purposes, the Hadoop JobTracker web interface visualisation tool was used. The JobTracker web interface provides a wealth of information on jobs and tasks that are running on the cluster as well as historical information on completed jobs (including ones that failed). The Analyse job history link on the Job details history page displays a summary of task performance statistics and details of individual task attempts can be extracted.

The complex queries including the D2C algorithm are implemented as MapReduce jobs. The D2C is heavily optimised to run in parallel. For a given D2C process, a number of mappers are spawned across the nodes. Each mapper works simultaneously on a set of timesteps from the complete datasets to compute results associated to mesh points in the continuum field. This amounts to the top-level parallelism present in the algorithm by processing independent timesteps based on where they are located across the nodes. Further parallelisation is exploited within each timestep as computation of results in the mesh can be decomposed into regions. Effectively, the D2C exploits two levels of parallelism: across nodes and within a node by leveraging multiple cores. The reduce phase performs a summary operation for, e.g., computing averages and storing results back into the database.

The two levels of optimisation that can be enabled during a D2C run are:

- **Op. 1:** the default optimisation is enabling parallelism across nodes, i.e., a parallel task is created for each timestep within the selected simulation time range. This may lead to multiple tasks running on: 1) a single node (if timesteps reside only on that node), 2) multiple nodes (if the data blocks are well distributed). In the second case, we may not be utilising all the cores on a single node.
- **Op. 2:** this optimisation enables the exploitation of nested parallelism with a task that processes a timestep and runs on a node by computing results associated with the output mesh in parallel.

The effect of optimisation level 1, which is increasing the number of nodes used to process the data, is shown in Figure 3 for the fluidised bed test case. The default optimisation (single processor per node) of the D2C algorithm is tested for various sizes of simulation data (20, 40 and 100 timesteps of data). In this case, each node only processes a single timestep of data on a single processor - parallelisation is achieved by distributing the timestep data across the nodes, where each node is given approximately one third of the data for processing. The results show slightly less than linear speedup, with a speed-up factor of 87% and measured memory ratio 0.925 showing efficient memory usage on one to three nodes. Due to the problem size, as only 100 or fewer timesteps of data (with each one just over 1 MB in size) are being processed, only three nodes are used as the job set-up time on each node and communication times start to become the limiting factors on performance for such a small problem.

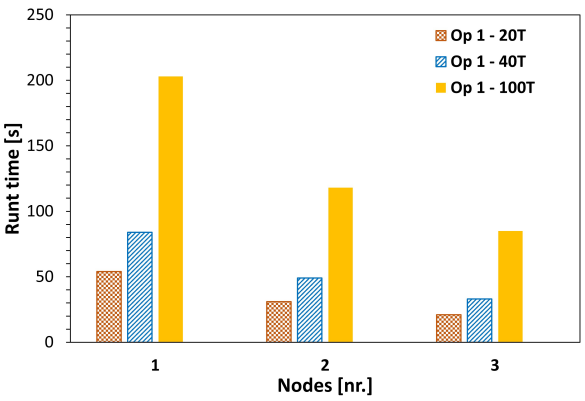


Figure 3. Performance benchmarks of initial D2C algorithm on small dataset (*Fluidised Bed* – see Table 1 for details)

While this level of optimisation shows good performance on relatively small datasets, it does not utilise the available resources fully. In order to quickly process much larger datasets, further optimisation of the D2C algorithm is required to exploit all available cores on a node. At a node level, data can be further decomposed into regions and results computed in parallel using multiple processor cores available on that node (Op. 2). The increase in performance from this further optimisation is illustrated by Figure 4, where the effect of node-level parallelisation and node usage is shown for the large dataset. The data falls into two distinct groups based on the level of optimisation used. Without node-level parallelisation (Op. 1), the processing runtimes using different amounts of level one optimisation (1, 2 &

10 nodes) are significantly higher as reflected by the three top data series. As the number of nodes utilised is increased, the total runtime begins to decrease significantly, to the point where running 10 timesteps of data across 10 nodes is only slightly slower than one timestep on one node.

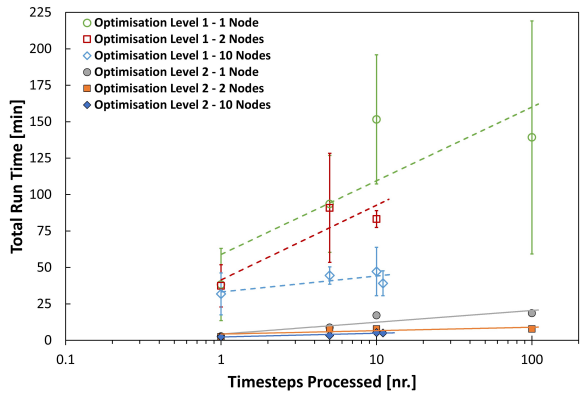


Figure 4. Performance of D2C algorithm on a large dataset (*Railway Embankment* – see Table 1 for details)

However, by enabling the second level of optimisation, the algorithm effectively takes advantage of all processor cores on the node and shared memory access of the timestep being processed and consequently reduces the time it takes to compute results for all data significantly. The second parallel optimisation is on average ten times faster than the first version across different simulations tested. Additionally, Figure 5 compiles the normalised runtime cost for both small and large datasets with level 2 optimisation over increasing number of nodes showing a high R^2 value with a consistent decrease in processing time per MB of data.

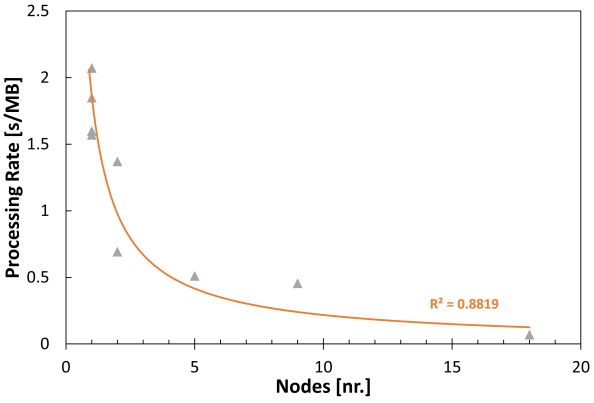


Figure 5. Performance of D2C algorithm for different sized datasets as runtime per MB of data processed

Sample Applications and Post-processed Results

Two specific use cases (one small dataset, one large dataset) have been considered for the evaluation of the VELaSSCo platform for DEM data. The smaller dataset is for a simple fluidised bed simulation while the large dataset is for a railway embankment simulation. The details of the use cases are presented in Table 1. Both use cases cover some of the more challenging aspects of dealing with DEM data: large amounts of particle and contact data and many timesteps.

Table 1. Use Case Details

	Fluidised Bed	Rail Embankment
Number of particles	11,880	207,440
Number of contacts	≈ 3,000	≈ 560,000
Number of timesteps	40,800	9,100
Simulation size	50.4 GB	541 GB
Timestep data size	1.12 MB	62 MB
Number of result variables	5 (2 scalars + 1 vector)	8 (2 scalars + 2 vectors)
Contact data processed	No	Yes

Use Case 1: Fluidised Bed

Over the years, industries have realised that fluidised systems offer many advantages, and as a result, fluidised bed processes have become commonplace in the pharmaceutical and other chemical industries. They are able to provide high levels of contact between a solid and a gas, which makes them extremely useful as driers; Fluidised-Bed Catalytic Cracking (FCC) is one of the most important and widely used refinery processes for converting low value, heavy oils into more valuable lighter products. They are also commonly used for particle coating due to both their high gas/solid surface area and high level of mixing.

The Fluidised Bed use case presented here (Figure 6) is the smaller of the two use cases, although it contains data for a very large number of timesteps. Due to the dynamic nature of fluidised beds, the simulation was sampled at a high frequency, and as a result, a total of 40,800 timesteps of data were generated. The results of interest in this simulation are mass, volume, velocity vector along with the contact force vector. The simulation was carried out on an HPC cluster using the molecular dynamics code LAMMPS[8].

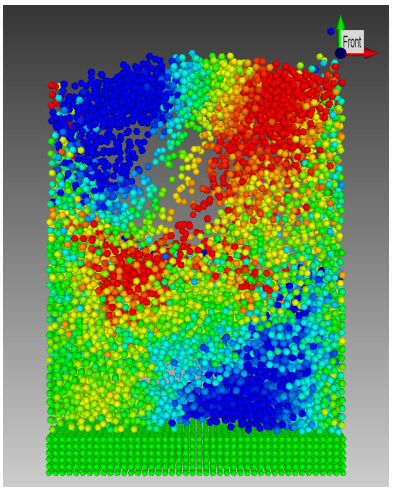


Figure 6. Visualised fluidised bed coloured by particle vertical velocity

Figure 6 shows a single timestep of the simulation visualised on the VELaSSCo platform using the iFX

visualisation client with the particle colour representing the vertical velocity (V_z). The clients also support the animation of many timesteps to easily visualise dynamic processes.

The platform provides the ability to query both the discrete particle data and the transformed continuum data. Particle-level information could be used to identify when particles experience specific zones or regimes within the bed, such as passing through a bubble or being at the free surface, by the changing velocity history. It may also be used to analyse the cyclic nature of the data and determine the frequency of any repeating phenomenon occurring in the simulation. Figure 8 shows the effect of temporal averaging on the evolution of the spatially averaged velocity magnitude for a chosen node which is at the centre of a horizontal plane through the bed at approximately one-quarter bed height.

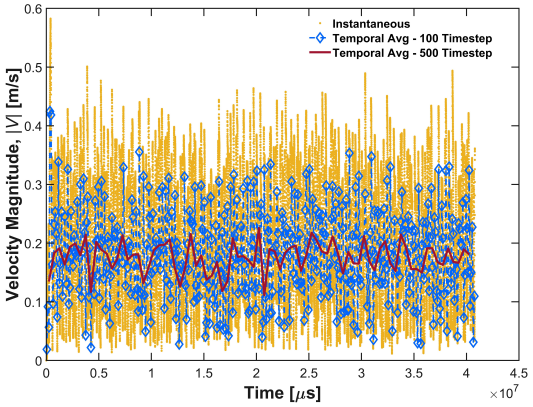


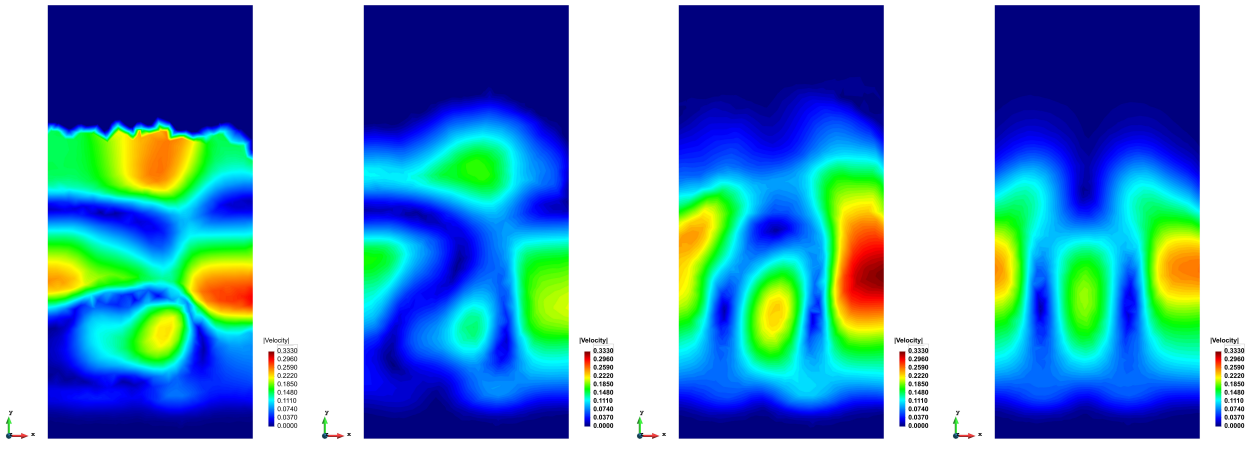
Figure 8. Effect of temporal averaging on the evolution of spatially averaged velocity magnitude at a fixed point in the bed for the full simulation duration

The combination of both spatial and temporal averaging can be more fully visualised in Figure 7 where the velocity field at a single timestep (at t=20.4 s) is compared with the computed spatial averages for three different temporal windows: 100, 500 and 40,800 timesteps or 100 ms, 500 ms and 40.8 s. The instantaneous result (Figure 7a) shows the locally averaged particle velocity at the particular instant of t=20.4 s in the simulation. The instantaneous velocity appears significantly different to temporally averaged results at the selected timestep, with particles close to the free surface being highlighted as some of the fastest moving. The results temporally averaged over 100 ms (Figure 7b) show significantly reduced velocities with four 'hotspot' zones. In comparison, the results temporally averaged over 500 ms (Figure 7c) show relatively low velocities in the upper region, suggesting that particles only intermittently enter this zone when they have a high velocity. Instead, the high-velocity zones are seen as vertical corridors pointing to a circulation pattern over the longer time frame. The final temporal window which averages the simulation data over a much longer duration of 40.8 seconds shows that the top of the bed typically has low particle velocities and the vertical circulation channels where high velocities are observed are now clearly visible. The similarity between the patterns observed in Figure 7c and Figure 7d suggests that some cyclic behaviour occurs in the bed, with 500 timesteps being large enough to capture at least one full cycle length.

Use Case 2: Railway Embankment

Significant demands are being placed on rail networks around the world, with growing traffic pushing the limits of the existing infrastructure. Extra traffic is accommodated through both higher train speeds and larger axle loads. Considering the naturally discrete, inhomogeneous structure that is a ballasted trackbed, DEM is ideally suited to study ballasted railway infrastructure. With advanced analysis tools, the stress distributions and deformation patterns that develop in ballasted track systems can be analysed, providing key scientific insights into ballasted tracks. However, in order for the simulations to provide sufficient insight, the simulations must be carried out on a large scale, which drives the need for a post-processing tool capable of dealing with the large volumes of data generated.

The simulation, which was carried out in the commercial code EDEM[38], is shown in Figure 9 and produced a dataset of 540 GB for the short duration investigated. The results processed and analysed with this dataset are mass, volume, translational and angular velocity vectors and the contact force vector.



(a) Instantaneous velocity

(b) Velocity temporally averaged over 100 timesteps

(c) Velocity temporally averaged over 500 timesteps

(d) Velocity temporally averaged over 40,800 timesteps

Figure 7. Comparison of temporally averaging window length on velocity magnitude in a fluidised bed centred on t=20.4 s

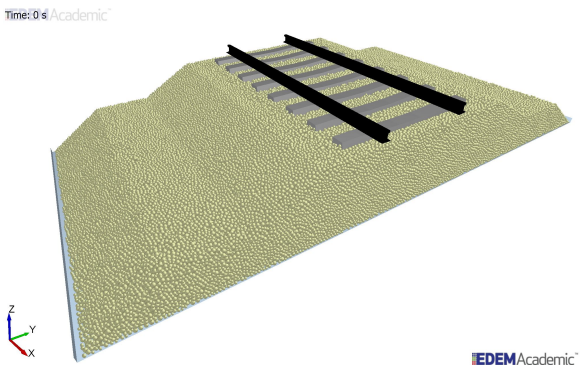
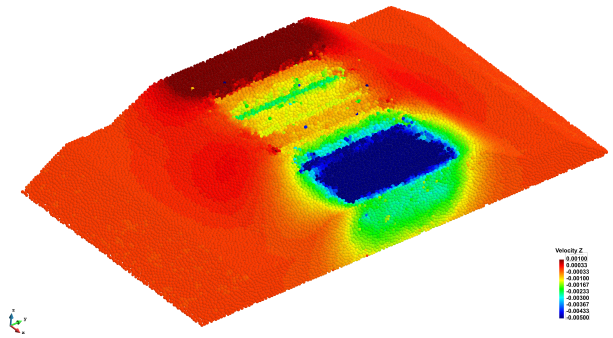
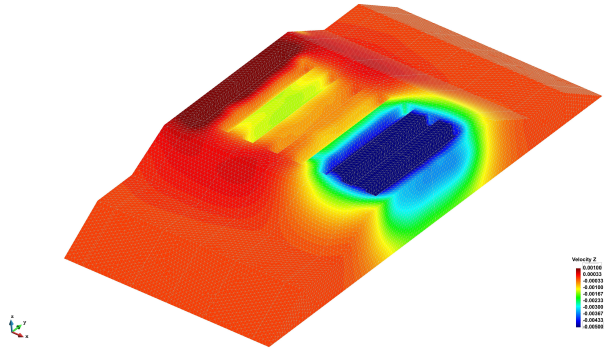


Figure 9. DEM simulation of a railway embankment



(a) Instantaneous particle vertical velocity



(b) Spatially averaged instantaneous vertical velocity

Figure 10. Vertical velocity in embankment at timestep t=5.17 s

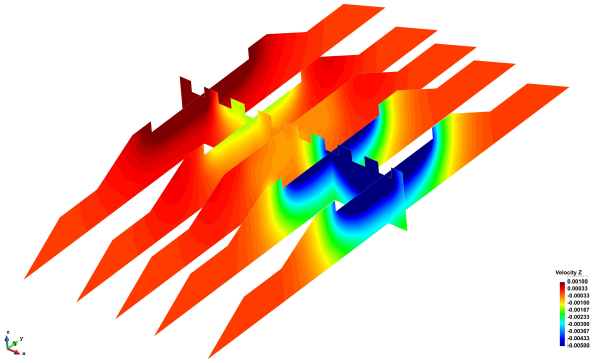


Figure 11. Cut planes in railway embankment showing vertical velocity at timestep t=5.17 s

The railway embankment simulations are carried out to study the effect of cyclic loading due to many trains passing, and as such, are long-duration simulations which generate large amounts of data. This use case examines the effect of ten cycles of train loading; results from the first cycle are presented here. The train is travelling from left-to-right (positive X direction) with a velocity of 70 km/h.

In Figure 10, the loading is applied by a two-axle bogey. The first axle of the bogey is just approaching the sixth sleeper while the second axle of the bogey is directly located on the second sleeper. Figure 10a shows the discrete velocities of every particle for a single timestep at t=5.17 s while Figure 10b shows the spatially averaged, coarse grained (Gaussian width of 2 particle diameters and cut-off factor of 3) vertical velocity results for the same timestep. The loaded bogey (axle pair) causes some downwards velocities in its immediate vicinity; however, the largest downward velocities are seen prior to the axle arriving (sleepers 7 & 8) due to the load being transmitted to the next sleepers through the rail. The largest upward velocities are seen after the axle has passed the sleepers, such as at the first sleeper which is unloaded. The railway embankment simulations are quasi-static scenarios where particle velocities and displacements are very low despite the large forces applied. Due to this, there are only subtle differences between the discrete particle velocities (Figure 10a) and the spatially averaged instantaneous velocities (Figure 10b), where the extreme local particle velocities which are not representative of the bulk response are averaged out.

The results in Figure 10 are only displayed on the outer skin of the 3D mesh. In order to further interrogate data we can use the functionality of the visualisation client to make several cut-planes through the 3D mesh to visualise and examine what is occurring internally in the embankment. Figure 11 shows the same vertical velocity information, but this time on cut-planes taken through the embankment at six locations: four cross-sections under the first, third, sixth and eighth sleepers, a cross-section through the middle of the embankment and a long-section along the centre line of the embankment. This allows the end-user to see the extent to which the embankment is affected internally from the loading.

While contour plots are very useful for understanding the overall picture, it can be difficult to quantitatively assess these results. In Figure 12 the vertical velocity profiles at two different depths for some of the cross-sections shown in Figure 11 are plotted, allowing the simple comparison to take place at predefined locations.

In addition to the ability to spatially and temporally average particle data, the Discrete2Continuum tool also processes contact data (where available) to compute the stress field, and other properties such as bulk density, momentum and kinetic stress. This provides a wealth of information with which the end user can gain insight into their simulation. The bulk density variation on three cut planes in the embankment is shown in Figure 13. The results show a significant variation in the bulk density in the bed, which is actually quite representative of the inhomogeneous nature of a real-life embankment. This allows the end-user to investigate how the particle packing has been formed and to identify possible ‘soft spots’ in the embankment, i.e., areas in which the packing density is reduced, and hence have a lower bulk stiffness than the surrounding areas. When used in conjunction with the DEM output of sleeper geometry

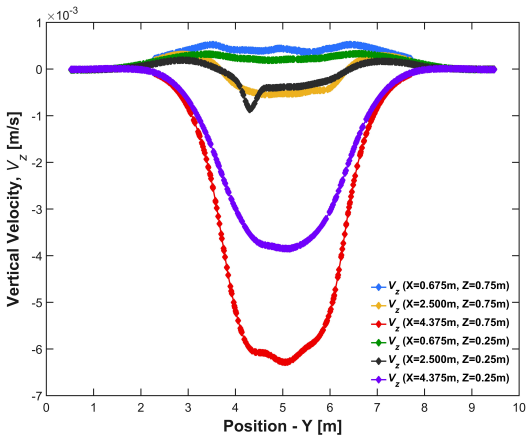
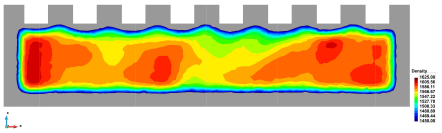
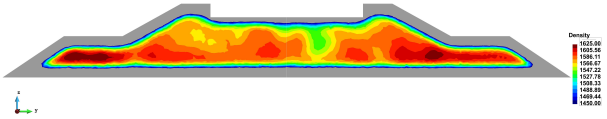


Figure 12. Vertical velocity profiles at two depths under sleeper 2 (X=0.675 m), Sleeper 7 (X=4.375 m) and mid-embankment (X=2.5 m) at timestep t=5.17 s

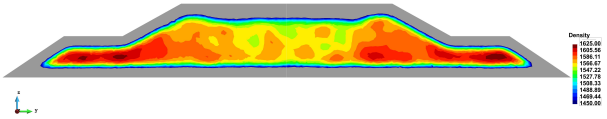
displacement, the relationship between deformation and packing arrangement can be investigated and used to explain phenomena such as excessive sleeper settlement. The grey areas in these figures represent areas where the computed density is artificially low close to the simulation boundary. This is an artefact of the averaging method employed in the analysis, as no boundary correction is currently applied for density.



(a) Density variation on long-section through embankment



(b) Density variation through embankment at 3rd sleeper



(c) Density variation through embankment at mid-distance

Figure 13. Density variation in embankment

The computed horizontal (S_{xx}) and vertical (S_{zz}) stresses are shown for a long-section through the centre of the embankment in Figure 14 and Figure 15, both of which can provide important information to the end-user. The stress field is typically of interest to the engineer, but unlike finite element simulations, DEM simulations do not provide this as a native result and this can be an expensive calculation in large models. The vertical stress, S_{zz}, is plotted in Figure 14 and is linked directly to the positions of the axle loads applied. The heaviest loaded axle is that at sleeper two (second void from left) which leads to the largest vertical stress developing under this location. The section of track between the two loaded sleepers experiences the largest stresses as the rails transmit some of the force into the intermediate sleepers helping to distribute the load more evenly. Although the first axle is approaching the sixth sleeper, its load is currently being distributed between the fifth and sixth sleeper. Under sleepers seven and eight, where the velocities of the particles are the highest, the stress levels remain low (mainly self-weight at this stage) until the axle load travels further along the track. Very low stresses, which relate only to material self-weight, are observed in the cribs between the sleepers.

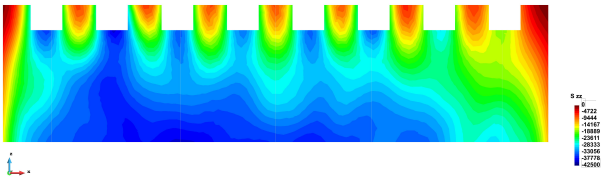


Figure 14. Vertical stress, S_{zz} in long-section, at t=5.17 s

In Figure 15 the highest longitudinal stresses, S_{xx}, are shown and it demonstrates a similar trend to that observed in the vertical stress. The largest horizontal stresses are observed between the second and third sleepers, which is where the heaviest axle load is positioned. Stresses are high in the same region as the vertical stress, although the horizontal stress is increasing in the area of high particle velocities under sleeper seven before the axle load has been applied directly to this sleeper. This information is particularly useful to the engineer as the stress levels will typically increase with higher speeds or higher axle loads. Using this simulation information, the safe loading limit of the embankment could be established or the ability of the underlying ground conditions to support such loads could be investigated.

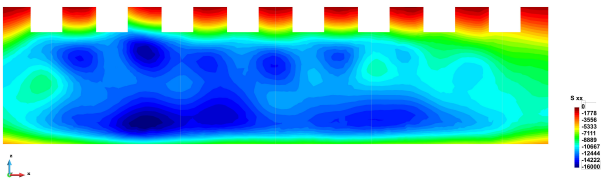


Figure 15. Horizontal stress, S_{xx}, in long-section at t=5.17 s

The VELaSSCo platform has been developed to store, manage and post-process very large simulation datasets. The results presented in the use cases so far have shown the post-processing capabilities of the platform for interrogating a single timestep of results. However, the platform also provides the ability to query the entire time range of the data, not just the currently loaded timestep. A more typical application of this capability is the creation of animations of the dataset to visualise the dynamic behaviour exhibited, such as the example of a Fluidised Bed[39] from Use Case 1.

Further utilising this ability, the end user can extract some results and see their variation over time such as in Figure 16, where the evolution of the vertical stress in the rail embankment at three points of interest for the first loading cycle is shown. While analysis like this is trivial for small datasets and could easily be accomplished on a conventional workstation, this sort of analysis would not be possible

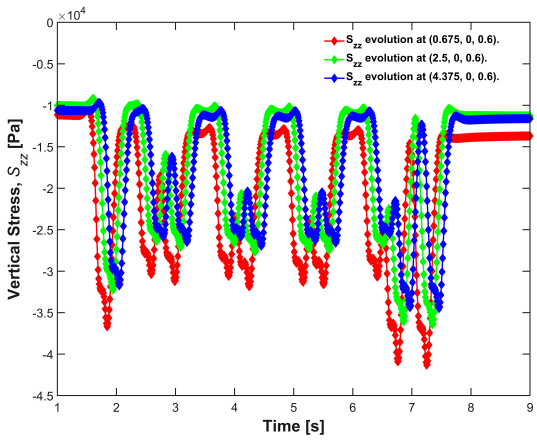


Figure 16. Vertical stress evolution at three locations in embankment

for large datasets on such a workstation due to the limited memory, storage space and computational power.

Conclusions

This paper has described the development of the VELaSSCo platform which is a powerful, flexible tool for the visualisation and analysis of large datasets. The platform is built on top of the open-source Hadoop software stack and utilises the Hadoop Distributed File System (HDFS) to safely and efficiently manage extremely large datasets. The architecture has been designed to allow fast processing of simulation data on the cluster before efficiently transferring the requested result back to the visualisation client on the desktop. Scalability was a key consideration for the architecture and the platform can be deployed on custom clusters consisting of several hundreds or thousands of nodes, and accommodate very large distributed datasets. Key to improved performance is an effective distribution strategy of data across the nodes, and a processing algorithm adapted to take advantage of this.

There are many DEM solvers currently freely available as open-source software, as well as many other commercial codes. In order to make the platform work with such a large group of software a file format for exporting DEM data was developed and this has been accepted as an extension to ISO 10303-209 *"Multidisciplinary Analysis and Design"* [35].

The Discrete2Continuum module is a tool that provides advanced analytics for DEM datasets, far beyond the simple visualisation and querying of particle properties. The module is a highly optimised toolbox that can analyse the particle data across different spatial and temporal length scales and provide key macroscopic properties such as stress, density and velocity. The distributed nature of the architecture means there is a significant performance increase for large datasets from adopting Big Data methodology for importing and storing raw data. The algorithm has been heavily optimised to run in parallel across both multiple cores on a node and multiple nodes on a cluster, which has led to significant decreases in run time from a simple implementation, as would be common on a conventional workstation.

A selection of use cases have been presented to show the VELaSSCo platform at work. These use cases highlight datasets that are typically difficult to process on a standard workstation due to their large size and memory footprint. Using the VELaSSCo platform, these datasets can be processed in the background and then streamed back to the visualisation client on the desktop for viewing of the results. The results highlight both the need and advantage of being able to carry out such spatial and temporal averaging on DEM datasets, as the end user is able to access revealing information that cannot be seen in the particle data alone.

Acknowledgements

The authors acknowledge the contribution made by the entire VELaSSCo consortium of seven partners to the development of the platform. The feedback of the user panel has been invaluable for guiding the development of VELaSSCo. This work has made use of the resources provided by the Edinburgh Compute and Data Facility (ECDF).

Funding

The development of the VELaSSCo platform was funded by the European Union's Seventh Framework Programme (FP7/2007–2013) under grant agreement n° 619439.

References

1. O'Sullivan C. *Particulate discrete element modelling: A geomechanics perspective*. First ed. Oxford, UK: Taylor & Francis, 2011.
2. Alder BJ and Wainwright TE. Phase transition for a hard sphere system. *Journal of Chemical Physics* 1957; 27: 1208–1209.
3. Alder BJ and Wainwright TE. Studies in molecular dynamics. I. General method. *Journal of Chemical Physics* 1959; 31(2): 459–466.
4. Cundall PA and Strack ODL. A discrete numerical model for granular assemblies. *Géotechnique* 1979; 29(1): 47–65.
5. O'Sullivan C. Particle-based discrete element modelling: Geomechanics perspective. *International Journal of Geomechanics* 2011; 11(6): 449–464.
6. Marigo M and Stitt EH. Discrete element method (DEM) for industrial applications: Comments on calibration and validation for the modelling of cylindrical pellets. *KONA Powder and Particle Journal* 2015; 32: 236–252. DOI:https://doi.org/10.14356/kona.2015016.
7. Zhu HP, Zhou ZY, Yang RY et al. Discrete particle simulation of particulate systems: Theoretical developments. *Chemical Engineering Science* 2007; 62(13): 3378–3396.
8. Plimpton S. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics* 1995; 117: 1–19.
9. Kloss C, Goniva C, Hager A et al. Models, algorithms and validation for opensource DEM and CFD–DEM. *Progress in Computational Fluid Dynamics* 2012; 12(2/3): 140–152. DOI: https://doi.org/10.1504/PCFD.2012.047457.
10. Itasca Consulting Group. PFC 2D and 3D discrete element (DEM) particle model simulation software, 2016. URL http://www.itascacg.com/software/pfc.
11. DEM Solutions Ltd. EDEM 2017 Release Announcement, 2016. URL https://www.edemsimulation.com/news/press-release/edem-just-became-a-lot-faster-discover-edem-2017/.

12. Cundall PA. A discontinuous future for numerical modelling in geomechanics? *Proceedings of the Institution of Civil Engineers - Geotechnical Engineering* 2001; 148(1): 41–47.
13. Vendruscolo M and Dobson CM. Protein dynamics: Moore's Law in Molecular Biology. *Curr Biol* 2011; 21(2): R68–R70.
14. Goldhirsch I. Stress, stress asymmetry and couple stress: from discrete particles to continuum fields. *Granular Matter* 2010; 12(3): 239–256.
15. Labra C, Ooi JY and Sun J. Spatial and Temporal Coarse-Graining for DEM Analysis. In *Powders & Grains 2013*, volume 1542. Sydney, Australia: AIP Publishing, pp. 1258–1261. DOI:10.1063/1.4812167.
16. Weinhart T, Labra C, Luding S et al. Influence of coarse-graining parameters on the analysis of DEM simulations of silo flow. *Powder Technology* 2016; 293: 138–148.
17. Clark AH, Mort P and Behringer RP. Coarse graining for an impeller-driven mixer system. *Granular Matter* 2012; 14(2): 283–288.
18. Marigo M, Cairns DL, Davies M et al. A numerical comparison of mixing efficiencies of solids in a cylindrical vessel subject to a range of motions. *Powder Technology* 2012; 217: 540–547.
19. VELaSSCo. Visualization for Extremely Large-Scale Scientific Computing, 2016. URL http://www.cimne.com/projects/velassco/index.html.
20. Janda A, Filippone G, Hanley KJ et al. Post-processing of large-scale DEM simulations for end-user analytics and visualisation. In *Proc NAFEMS UK Conference 2016*. Telford, UK, pp. 97–100.
21. Zienkiewicz OC, Taylor RL and Zhu JZ. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier LTD, Oxford, 2013. ISBN 1856176339.
22. Zienkiewicz OC, Taylor RL, Fox DA et al. *The Finite Element Method for Solid and Structural Mechanics*. Elsevier LTD, Oxford, 2013. ISBN 1856176347.
23. Zienkiewicz OC, Taylor RL and Nithiarasu P. *The Finite Element Method for Fluid Dynamics*. BUTTERWORTH HEINEMANN, 2013. ISBN 1856176355.
24. Batchelor GK. *An Introduction to Fluid Dynamics*. Cambridge Mathematical Library, Cambridge University Press, 2000. ISBN 9780511800955. DOI:10.1017/CBO9780511800955.
25. Versteeg H and Malalasekera W. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. New York, 1995. ISBN 9780470235157.
26. Anderson JD, Degroote J, Degrez G et al. *Computational Fluid Dynamics: An Introduction*. 3 ed. A von Karman Institute book, Springer Berlin Heidelberg, 2009. ISBN 9783540850564. DOI:10.1007/978-3-540-85056-4.
27. Fenton N and Bieman J. *Software metrics: A rigorous and practical approach*. Third ed. Boca Raton, FL, USA: CRC Press, 2014.
28. CIMNE. GiD, 2017. URL http://www.gidhome.com/.
29. Fraunhofer IGD. iFX, 2017. URL http://www.i-fx.net/.
30. Akenine-Moller T, Moller T and Haines E. *Real-Time Rendering*. 2nd ed. Natick, MA, USA: A. K. Peters, Ltd., 2002. ISBN 1568811829.
31. Sigg C, Weyrich T, Botsch M et al. Gpu-based ray-casting of quadratic surfaces. In *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*. SPBG'06, Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. ISBN 3-905673-32-0, pp. 59–65. DOI:10.2312/SPBG/SPBG06/059-065.
32. Luft T, Colditz C and Deussen O. Image enhancement by unsharp masking the depth buffer. In *ACM SIGGRAPH 2006 Papers*. SIGGRAPH '06, New York, NY, USA: ACM. ISBN 1-59593-364-6, pp. 1206–1213. DOI:10.1145/1179352.1142016.
33. Webb R and Gigante M. Using dynamic bounding volume hierarchies to improve efficiency of rigid body simulations. In *Proceedings of the 10th International Conference of the Computer Graphics Society on Visual Computing : Integrating Computer Graphics with Computer Vision*. CG International '92, New York, NY, USA: Springer-Verlag New York, Inc. ISBN 0-387-70103-6, pp. 825–842.
34. Kajiya JT. The rendering equation. *ACM SIGGRAPH Computer Graphics* 1986; 20(4): 143–150. DOI:10.1145/15886.15902.
35. for Standardization (ISO) IO. Industrial automation systems and integration – Product data representation and exchange – Part 209: Application protocol: *Multidisciplinary analysis and design*, 2014.
36. VELaSSCo. VELaSSCo DEM Model to be part of ISO standard, 2016. URL http://www.cimne.com/projects/velassco/content/velassco-dem-model-be-part-iso-standard.html.
37. Particle Analytics Ltd. Particle Analytics v0.11.17, 2016. URL http://particle-analytics.com/.
38. DEM Solutions Ltd. EDEM 2.7.1, 2015.
39. VELaSSCo. Particle dynamics in a fluidised bed. online, 2016. URL http://www.cimne.com/projects/velassco/content/new-video-particles-animation.html.